Weekly Report

January 14, 2018

1 Work

1.1 降维

本周完善了论文的摘要和第一节到第四节，剩下一个实验章节要等程序写好后，再生成一些性能比较的图。

1.2 工作进度

Table 1: 工作进度

TASK	PROGRESS	DATE
dimension reduction	1)节省内存；2)写论文	1.17
location2vec专利		
*2Vec survey		2.20

2 Paper Reading

2.1 Attributed Network Embedding for Learning in a Dynamic Environment

文章提出了一个新方法，DANE，用于学习节点属性动态变化的网络表达。

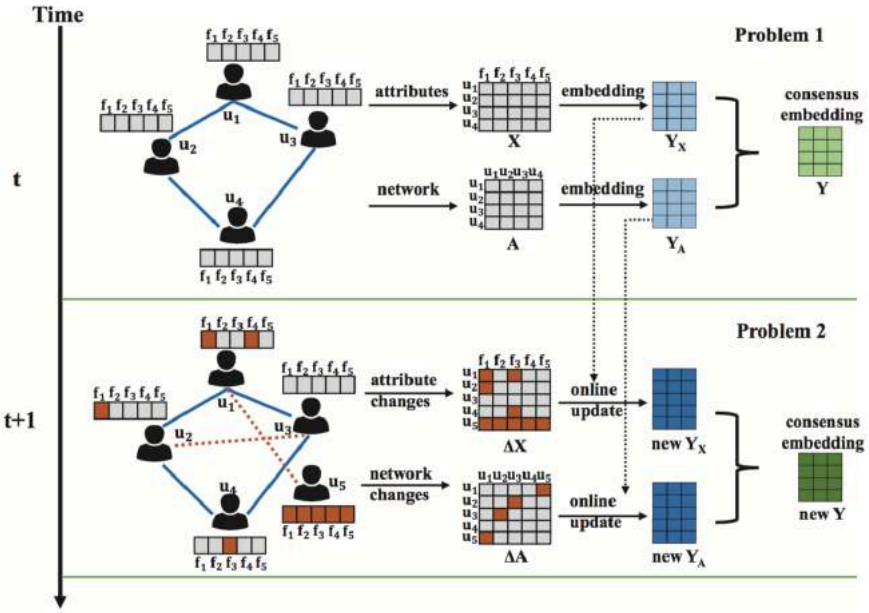


Figure 1: DANE

2.2 Word Embeddings, Sense Embeddings and their Application to Word Sense Induction

在词嵌入过程中，同一个单词可能会有不同的含义，比如苹果，所以嵌入的时候要考虑上下文语境。

2.3 Attributed Signed Network Embedding

和第一篇是类似的作者，这里考虑了网络嵌入时节点的属性和边的属性。

2.4 A Latent Space Approach to Dynamic Embedding of Co-occurrence Data

也是一篇考虑动态嵌入的文章，作者并没有说输入的数据是网络形式，但是实验中使用了NIPS论文信息，所以可以看作是是支持网络数据的方法。

2.5 Knowledge Graph Embedding: A Survey of Approaches and Applications

一篇知识图谱嵌入的工作，总结了目前的一些方法，比如TransA，TransM，TransD，TransF，还有使用神经网络的方法（提前将单词转换为向量做训练）。

同时讲诉了应用场景，包括边预测，节点分类，判断实体是否为同一个，其他的应用还包括关系抽取，问答和推荐。知识图谱上的应该感觉会比普通的网络更多一些。

Dimensional Reduction

Michael Shell, *Member, IEEE,* John Doe, *Fellow, OSA,* and Jane Doe, *Life Fellow, IEEE*

Abstract—Visualizing high dimension data is an essential task in visualization and machine learning. Over the past decades, a large number of dimensional reduction methods have been proposed to generate 2D embedding for data analysis. t-Distributed Stochastic Neighbor Embedding (tSNE) is one of the most popular techniques, which achieve the outstanding performance. However, most existing embedding approaches takes too much time on large-scale data. We propose an efficient dimensional embedding method with sub-linear computation complexity by sharing the gradient between adjacent data points and reducing the number of gradient that needed to be computed. Our experiments on various large-scale data sets show the significant acceleration of our method with a comparable embedding quality with LargeVis.

Index Terms—Dimension reduction, High-dimensional, KNN Graph, visualization.

1 INTRODUCTION

REPRESENTING high-dimensional data via compacting vectors via dimensional reduction has been widely studied in several fields of science. Visualization is an important part of data analysis. Analysts are able to get an overview of data distribution and generate hypotheses before data process. Since human perception is confined to low dimensional space, we usually project high dimensional data into low dimensional space. Each data object is represent as a object in the low dimensional space in this way. In low-dimensional space, adjacent data objects are tend to share similar attributes and dissimilar objects are far away from each other. Data objects in low dimensional space can be visualized as scatter plot first. Other visualization approaches such as scalar field and heat map encode the quality or uncertainty of embedding for further exploration. The applications of dimensional embedding include visualization [22], deep learning [13], life science [19], and network analysis [2], [17], etc.

Generally, dimension reduction technologies convert the high dimensional data set $X = \{x_1, x_2, ..., x_N, x_n \in \mathcal{R}^D\}$ into low dimensional data set $Y = \{y_1, y_2, ..., y_N, y_n \in \mathcal{R}^2\}$ for visual analysis. Over the past decades, a large number of dimensional reduction methods have been proposed. The classical techniques include principal component analysis (PCA) [12], linear discriminant analysis (LDA) [4] and multidimensional scaling (MDS) [16]. To preserve the local properties of the data, some new dimensional reduction methods have been proposed, such as locally linear embedding (LLE), locality preserving projections (LLP) and neighborhood preserving embedding [7].

Recently, stochastic neighbor embedding (SNE) based dimensional reduction methods have achieved outstanding performance. Maaten proposed t-distributed stochastic neighbor embedding (t-SNE) [18] to solve the crowding problem. However, the computational complexity of t-SNE $O(N^2)$ scales with the size of data set, it is time consuming to visualize large scale data sets. To accelerate t-SNE, Mattern [28] employed KNN graph and quadtree to speed up the gradient computation with the complexity of $O(NlogN)$. Recently, LargeVis [25] optimizes the object function with negative sampling method, which reduce the time complexity in terms of the number of data items $O(N)$. During the optimization process, we find that LargeVis takes a lot of time merge clusters with the same label (see Fig.?) due to weak connection between two clusters.

In this paper, we develop a sublinear algorithm to accelerate the computation of LargeVis. We accelerate approximated k-nearest neighborhood (KNN) graph construction by leveraging an efficient algorithm proposed in EFANNA [5]. EFANAA generates an initial graph by KD-tree and refines the graph with the NN-descent. (ł) We expand the KNN graph by preserving the neighbor candidates pool of NN-descent, which results a larger graph whose accuracy of the top neighbors is very high and the accuracy of neighbors from candidates pool is a litter lower.

Then, we modify the optimization by approximating the gradient based on the cluster level. Inspired by BH-SNE [28] which approximates the gradient by sharing distance in the same cell, our method reduce the number of gradient that needed to be computed by sharing the gradient between nearby data points. First, we enlarge the attractive forces of the gradient to speed up the aggregation between similar points. Second, we divide the two-dimensional data points into clusters which share similar gradient. Third, we can compute only once for each cluster in each iteration.

Furthermore, our method is performed on various large scale data sets and achieves significantly acceleration over previous work. The experimental results demonstrate the efficiency of our approach.

In summary, our contribution includes:

- We propose a novel dimensional embedding method, which scales sub-linearly with the number of input data. We approximate the optimization by sharing the gradient between adjacent data points.
- We conduct experiments on various large scale data

- *M. Shell was with the Department of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA, 30332. E-mail: see http://www.michaelshell.org/contact.html*
- *J. Doe and J. Doe are with Anonymous University.*

Manuscript received April 19, 2005; revised August 26, 2015.

sets. The results show the efficiency of our method with a comparable embedding quality with previous work.

The remainder of the paper is organized as follows. In section 2, we review the related embedding approaches. We introduce our method in Section 3. The experimental results are presented in Section 4. Finally, we draw conclusions in Section 5.

2 Related work

Dimensional reduction approaches reduce the number of input variables to accelerate the computation or project high-dimensional data into low-dimension space for visual exploration and inherent structure reveal. The dimensional reduction approaches are usually classified into two categories: linear embedding and nonlinear methods.

Linear dimensional reduction method project high dimension data based on a linear product transformation. The distance relationship among data point in original space will be preserved to low dimensional space. Principal component analysis (PCA) [12] is the most popular and widely used method, which minimizes reconstruction error between high-dimension data and high-dimension data. Multidimensional scaling (MDS) [16] is another classical dimensional reduction technique. The goal of MDS is preserving pairwise distances between two space. MDS only requires the pairwise distance which is useful in some situations. In addition, it is proofed that MDS is equivalent to PCA in Euclidean space [3]. Likewisethe aim of Sammom mapping [23] is to minimize the distance error which is optimized by steepest descent procedure. When data has associated class labels, linear discriminant analysis (LDA) [4] is employed to reveal label information. LDA maximize the distance between different clusters and minimize the distance between data point in each cluster.

Most approaches such as PCA and MDS work well for data on a linear subspace, but they fail to detect nonlinear manifold in high-dimensional space. To capture non-linear structure, nonlinear dimension reduction algorithm always employ non-linear distance metric or preserve the local structure. Isomap [27] estimates the geodesic distance instead of Eucliden distance to minimize the pairwise distance error. Geodesic distance is more suitable to reflect nonlinear low dimension manifold such as the "swiss roll" data set. Other nonlinear embedding methods attempt to preserve the local structure: nearby points in high dimensional space remain nearby in the low dimension space. The basic idea of locally linear embedding (LLE) [24] is reconstructing data points with the linear combination of neighbors in high dimension space and minimizing the reconstruction error in low-dimensional space. Both Laplacian Eigenmap [1] and locality preserving projections (LPP) [8] attempt to minimize the distance between nearby points. LPP assumes that low dimension embedding is generated by a linear transformation. Therefore, LPP is suitable for new test points. Neural networks such as self-organize map [15] and autoencoder [10] can also be employed to reduce the dimension of data.

Unlike with previous methods, stochastic neighbor embedding [9] based approaches use probability rather than the distance to measure the similarity among the data points. The goal of these approaches is to minimize the Kullback-Leibler distance between two probability distributions on high dimension space and low dimension space. t-distributed stochastic neighbor embedding (SNE) is proposed to solve the crowing problem [18]. t-SNE shows its significant advantages in generating low-dimensional embedding. However, it is time consuming to visualize larger data sets with $O(N^2)$ computational complexity. To solve this issue, Maaten proposed Barnes-Hut-SNE (BH-SNE) [28], which ignore the edges with small probability, employ k-nearst neighborhood graph to estimate the rest probability and approximate distance in low dimension space by quadtree. Kim et. al. introduced an efficient version, called pixel-aligned SNE (PixelSNE) [14]. PixelSNE limits quadtree depth based on screen resolution which guarantees the minimum cell size is not smaller than a pixel. Largevis [25] employs a efficient algorithm to construct a k-nearest neighbor graph and speed up the optimization by negative sampling [20] and edge sampling [26] techniques.

3 Preliminaries

The family of SNE based approaches usually minimize the distance between two probability distributions on high dimensional space and low dimensional space. Due to the relatively high probability of adjacent data points, the optimization function will preferentially maintain the distance between adjacent data points The pipeline of existing algorithms is divided into three parts:

- **Input similarity computation**: compute input similarity probability between data points in high dimensional space;
- **Output similarity computation**: measure output similarity probability between data points in low dimensional space;
- **Optimization**: minimize the distance between two probability distributions.

In this section, we introduce the connection between existing methods (t-SNE and LargeVis).

3.1 t-Distribute Stochastic Neighbor Embedding

t-SNE minimizes the Kullback-Leibler divergences between two probability distributions of high-dimensional data and low-dimensional data.

Input similarity computation Mathematically, the probabilities between data points in high-dimensional space are defined as follows:

$$\begin{aligned} p_{j|i} &= \frac{exp(-d(x_i,x_j)^2/2\sigma_i^2)}{\sum_{k\neq i} exp(-d(x_i,x_k)^2/2\sigma_i^2)} \\ p_{i|i} &= 0 \\ p_{ij} &= \frac{p_{j|i}+p_{j|i}}{2N} \end{aligned}$$

where $d(x_i, x_j)$ is Euclidean distance between high dimension data x_i and x_j, σ_i is the variance of Gaussian distribution on data x_i and N is the data size.

Output similarity computation A heavy-tailed distribution is employed to measure the probabilities in low-dimensional space:

$$q_{ij} = \frac{(1+d(y_i,y_j)^2)^{-1}}{\sum_{k\neq l}(1+d(y_i,y_j)^2)^{-1}} = \frac{(1+d_{ij}^2)^{-1}}{Z}$$

TABLE 1
Summary of t-SNE based dimensional reduction methods.

	Attractive Force			Repulsive Force			Optimization
	Formula	Time Complexity	Number	Formula	Time Complexity	Number	
t-SNE	$\frac{q_{ij}(y_i-y_j)}{1+d_{ij}^2}$	N^2D	N	$\frac{q_{ij}(y_i-y_j)}{1+d_{ij}^2}$	N^2d	N	Batch
BHSNE	$\frac{q_{ij}(y_i-y_j)}{1+d_{ij}^2}$	kND	k	$\frac{q_{i,cell}(y_i-y_j)}{1+d_{i,cell}^2}$	$N\log(N)d$	N	Batch
LargeVis	$\frac{p_{ij}(y_i-y_j)}{1+d_{ij}^2}$	kND	k	$\frac{\gamma(y_i-y_{j_k})}{d_{ij}^2(1+d_{ij}^2)}$	NMd	M	Mini-batch
Ours	$\frac{p_{ij}(y_i-y_j)}{1+d_{ij}^2}$	kND	k	$\frac{\gamma(y_i-y_{j_k})}{d_{ij}^2(1+d_{ij}^2)}$	NMd/k	M	Mini-batch

where Z denotes the normalization term. For the sake of simplicity, we denote d_{ij} as the Euclidean distance between y_i and y_j.

Optimization The low-dimensional representations are learned by minimizing the Kullback-Leibler divergences between two probability:

$$C = KL(P||Q) = \sum_{i \neq j} p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

The gradient of the of the cost function C with respect to y_i is given by

$$\frac{\partial C}{\partial y_i} = \sum_{j \neq i} 4(p_{ij} - q_{ij})(1 + d_{ij}^2)^{-1}(y_i - y_j)$$

We split the search direction of the negative gradient into two parts:

$$\begin{aligned} dy_i &= -\frac{\partial C}{\partial y_i} \\ &= 4(-\sum_{j \neq i} p_{ij} q_{ij} Z(y_i - y_j) + \sum_{j \neq i} q_{ij}^2 Z(y_i - y_j)) \\ &= 4(-\sum_{j \neq i} F_{ij}^{attr} + \sum_{j \neq i} F_{ij}^{rep}) \\ F_{ij}^{attr} &= \frac{p_{ij}(y_i - y_j)}{1 + d_{ij}^2} \\ F_{ij}^{rep} &= \frac{q_{ij}(y_i - y_j)}{1 + d_{ij}^2} \end{aligned}$$

where F_{ij}^{attr} denotes the attractive force between y_i and y_j and F_{ij}^{rep} represents the repulsive force. t-SNE needs to compute input similarity p of attractive force in $O(N^2D)$ and output similarity q of repulsive forces with computational complexity of $O(N^2d)$.

Note that each data point in t-SNE receives $N-1$ attractive and $N-1$ repulsive forces from all the remaining data points. We need to compute calculate the gradients for all points and perform just one update for the entire data set. This batch gradient descent can be very slow on large-scale data sets.

3.2 Barnes-Hut-SNE

BH-SNE employs KNN graph and quadtree to accelerate the computation of attractive and repulsive forces.

Input similarity computation According to the gradient of t-SNE, computing the input similarities require $O(N^2d)$ which is too expensive. Recent approaches like BH-SNE and LargeVis employee k-nearest neighbor graph to approximate input similarities. The input similarities are redefined as:

$$p_{i|j} = \begin{cases} \frac{exp(-d(x_i,x_j)^2/2\sigma_i^2)}{\sum_{k \neq i} exp(-d(x_i,x_k)^2/2\sigma_i^2)} & \text{if } j \in N_k(y_i) \\ 0 & \text{otherwise} \end{cases}$$

Input similarity computation The idea of BH-SNE is that adjacent data points have similar distances to distant data point. Therefore, data points inside the same cell of quadtree can share distance.

Optimization BH-SNE only measure attractive force between k-nearest neighbors which reduce the complexity to $O(kND)$.

$$F_{ij}^{attr} = \begin{cases} \frac{p_{ij}(y_i-y_j)}{1+d_{ij}^2} & \text{if } j \in N_k(y_i) \\ 0 & \text{otherwise} \end{cases}$$

Additionally, BH-SNE approximate the repulsive forces in $O(N\log(N))$ by assigning the same distance to points in the same cell of quadtree.

$$F_{ij}^{rep} = \begin{cases} \frac{q_{ij}(y_i-y_j)}{1+d_{ij}^2} & \text{if } d_{ij}^2 < \theta \\ \frac{q_{i,cell}(y_i-y_{cell})}{1+d_{i,cell}^2}, j \in cell & \text{otherwise} \end{cases}$$

Note that each data point in BH-SNE receives k attractive and $N-1$ repulsive forces. However, the object function is still optimized based on batch gradient descent.

3.3 LargeVis

LargeVis adopts KNN graph and negative sampling method to accelerate optimization based on mini-batch gradient descent.

Input similarity computation Since constructing exact KNN graph is time consuming, LargeVis proposes an efficient approximate k-nearest neighbor graph construction method. First, an initial k-nearest neighbor graph is constructed based on random projection trees. Then, LargeVis refines the k-nearest neighbor graph by exploring the neighbor's neighbor to improve the accuracy.

Optimization Though the object function of LargeVis is defined on a graph, we discuss the optimization from the perspective of Kullback-Leibler divergence to be consist with the previous method. We can use the Kullback-Leibler divergence as loss function:

$$\begin{aligned} C = KL(P||Q) &= \sum_i \sum_j p_{ij} \log \frac{p_{ij}}{q_{ij}} \\ &= \sum_i \sum_j p_{ij} \log p_{ij} - p_{ij} \log q_{ij} \end{aligned}$$

Note that the first part of this equation is a constant. Therefore, minimizing the Kullback-Leibler divergence is equal to maximizing the following cost function:

$$\begin{aligned} \min \quad C \Leftrightarrow \quad & min \quad \sum_i \sum_j \quad p_{ij} \log p_{ij} - p_{ij} \log q_{ij} \\ \Leftrightarrow \quad & max \quad \sum_i \sum_j \quad p_{ij} \log q_{ij} \end{aligned}$$

Inspired by the negative sampling techniques, LargeVis randomly selects two connected points (y_i and y_j) in KNN graph and samples M negative points (y_{j_k}) for optimization:

$$O = \sum_{i,j} p_{ij} [\log q_{ij} + \sum_{k=1}^{M} \gamma \log(1 - q_{ij_k})]$$

The gradient of y_i is given by:

$$\begin{aligned} \frac{\partial O}{\partial y_i} &= -2F_{ij}^{attr} + \sum_{k=1}^{M} \gamma F_{ij_k}^{rep} \\ \frac{\partial O}{\partial y_j} &= 2F_{ij}^{attr} \\ \frac{\partial O}{\partial y_{j_k}} &= -2F_{ij_k}^{rep} \\ F_{ij}^{attr} &= \frac{p_{ij}(y_i - y_j)}{1 + d_{ij}^2} \\ F_{ij_k}^{rep} &= \frac{\gamma(y_i - y_{j_k})}{d_{ij_k}^2 (1 + d_{ij_k}^2)} \end{aligned}$$

Note that each data point in LargeVis receives 1 attractive force and M repulsive forces. The object function is optimized for every mini-batch of $M+2$ samples including two connected points and M negative points.

4 Our Method

In this section, we present an accelerated algorithm of LargeVis, which is faster than LargeVis by 5 times. We employ a faster KNN graph construction technique from EFANNA to accelerate the graph construction. In addition, we approximate the gradient to speed up the calculation of attractive and repulsive forces.

4.1 Input Similarity Computation

Since the similarities between near neighbors are sufficient to preserve the relationship in high dimensional space and the similarities between dissimilar points are nearly infinitesimal [18], we can compute the similarities to k-nearest neighbors for each point. In practice, we employ the KNN graph construction part of EFANNA [5] to accelerate the graph construction. EFANAA generates an initial graph by KD-tree and refines the graph with the NN-descent. The basic idea behind this approach is "a neighbor of a neighbor is also likely to be a neighbor". A-tSNE [22] shows that even KNN graph with low precision can preserve the overall clustering of data set. Therefore, we expand the KNN graph by preserving the neighbor candidates pool of NN-descent. The constructed graph has a high accuracy for its top neighbors. And other neighbors from candidates pool may have a litter lower accuracy. For instance, we can build a 10-NN graph whose accuracy is almost 100%, and expand it to 100-NN without additional time spent.

4.2 Optimization

The implementation of LargeVis employs asynchronous gradient descent to accelerate the optimization process. As discussed in section 3.3, each thread updates the positions of $M+2$ data points for each iteration. However, LargeVis suffers from two main drawbacks. First, LargeVis samples new negative points for each iteration, resulting in frequent memory access and additional computation. Second, as shown in Figure ?, it is time consuming to merge groups due to the weak connection between groups. Whenever a data point moves to another cluster with a small step, then it will be attracted back to the cluster it belongs.

We first discuss how to reduce the number of negative points to be sampled. The basic idea is sharing the negative point and positive point between similar points [11]. Fig.? shows the original process of LargeVis. For each data point, we need to sample a new positive point and M new negative points. It is straightforward to sample these points once and can be used repeatedly for gradient calculation for several times (see Fig.?).

To solve the second issue, the solution is to move all data points in the group together. We first assume that data points in the group are similar to each other, which means they have similar p_{ij}. Then, we can share positive and negative points for the group. Next assumption is that data points in the group are close to each other. The distances d_j and d_{j_k} from these data points to positive and negative points can be approximated as a random point to positive and negative points. Therefore, the attractive forces and repulsive forces are comparable for each point.

In summary, we can approximate the gradient of the whole group together only if they are similar in high dimensional space and close in low dimensional space. Computing the gradient once and assigning the gradient to other data points in the same group will accelerate the optimization process significantly.

For example, given two points (x_{j1} and x_{j2}) which are neighbors in the low dimensional space. First, y_{j1} and y_{j2} can share the same negative samples y_{j_k}. Therefore, they will have similar repulsive forces. If two points are also close in high-dimensional space. Then, their attractive forces are supposed to be similar due to comparable input similarity.

The whole optimization process includes three stages:

- **Generate groups** At first, all data points are initialized randomly. Similar data will group into small clusters during the early optimization process. After that, we generate groups where data points are similar in high dimensional space and close in low dimensional space. We first employ quadtree to divide the low dimensional space and adopt Girvan-Newman algorithm to group similar points based on KNN graph.
- **Merge groups** In this stage, we approximate the gradient to update the position at group level. It is more efficient to merge two similar groups by moving the entire group instead of moving each point. When we compute the gradient for a data object, we assign the gradient to the whole group which the data object belongs to. Similar groups will merge into a large group within several steps.

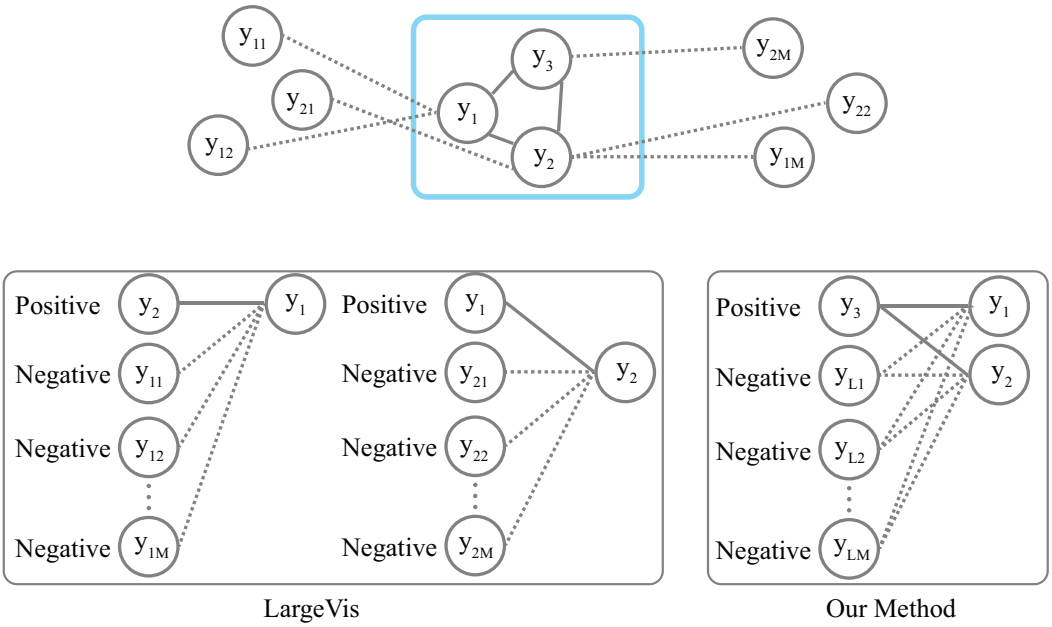


Fig. 1. The KNN graph is constructed on high dimensional space. The solid line represents edge which connected to positive point and the dashed line encodes the negative edge. Each data point is attracted by positive points and repulsed by the negative points. LargeVis sample new positive and negative points for each point. Our method generates these point once for each group sharing the negative point and positive points.

- **Refine layout** We stop moving group and refine the embedding result at data point level.

4.2.1 Generate groups

To generate data point groups, we first attract similar points to merge and then divide data points into groups which meets two conditions: 1) data points in the same group are close to each other; 2) data points in the same group are similar to each other.

Merge similar points During the early optimization process, the similar data points (with the same labels) tend to group into small clusters firstly. Then similar small clusters move to each other into large cluster. However, as mentioned in BH-SNE, it is much harder for the optimization to find a good embedding in later stages, especially for large-scale data set. In our experiments, we found that merging small clusters with the same labels is time consuming when there is another cluster lies between. The central cluster will push small clusters away since they are not close in the high dimensional space (see Figure ?). Therefore, the solution to this problem is enlarging the attractive forces between similar points. Previous works such as t-SNE and BH-SNE multiplied p_{ij} by a constant $\alpha(= 4\ or\ 12)$ to attract similar points into one group. Our work employ a similar strategy by decrease γ of repulsive forces in LargeVis. In practice, we fix $\gamma = 1$ for the first 10% optimization process and set $\gamma = 7$ in the remaining process.

Divide data To split low dimensional data points, we first employ a quadtree with max cell opacity of 100. The data points in each cell of the quadtree are close to each other (first condition). Then, we generate a graph within each cell based on KNN graph connection and employ a community detection algorithm, called Girvan-Newman [6], to further divide the cell into groups. The data points in each group are connected by edges in the KNN graph. Therefore, each group meets the second condition.

We compute a average movement distance after each 1% process:

$$Dis^j = \frac{1}{N}\sum_{i=1}^{N}(y_i^j - y_i^{j-1})$$

$$per = Dis^j / \sum_{i=1}^{5} Dis^{j-1}$$

We start dividing the data when their position are stable: $0.98 < per < 0.99$.

4.2.2 Merge groups

In this stage, we approximate the gradient computation of data points in the same group to accelerate the optimization. We perform quadtree and Girvan-Newman algorithms to generate $|G|$ clusters $G = G_1, G_2, ..., G_K$, where each group G_i contains in data points: $G_i = \{y_{i1}, y_{i2}, ..., y_{in}\}$. Then we are able to maximize the object function with $|G|$ groups.

We compute the gradient dy of all data points for i-th group and move the along the gradient:

$$y_i^{new} = y_i^{old} + \alpha\frac{\partial O}{\partial y_i}$$

First, we share the positive and negative points in the gradient of all data points:

$$y_i^{new} = y_i^{old} + \alpha(-2F_{ij}^{attr} + \sum_{k=1}^{M}\gamma F_{ij_k}^{rep}),\ \forall y_i \in G_i$$

Second, according to the property of groups how we generated, we approximate the attractive force and repulsive

forces of gradient by randomly selecting $p_{lj}(\approx p_{ij})$, $d_{lj}(\approx d_{ij})$ and $d_{lj_k}(\approx d_{ij_k})$ of data point $y_l \in C_i$:

$$\begin{aligned} F_{ij}^{attr} &= \frac{p_{ij}(y_i - y_j)}{1 + d_{ij}^2} \approx \frac{p_{lj}(y_l - y_j)}{1 + d_{lj}^2} = F_{lj}^{attr} \\ F_{ij_k}^{rep} &= \frac{\gamma(y_i - y_{j_k})}{d_{ij_k}^2(1 + d_{ij_k}^2)} \approx \frac{\gamma(y_l - y_{j_k})}{d_{lj_k}^2(1 + d_{lj_k}^2)} = F_{lj_k}^{rep} \\ y_i^{new} &\approx y_i^{old} + \alpha(-2F_{lj}^{attr} + \sum_{k=1}^{M} \gamma F_{lj_k}^{rep}),\ \forall y_i \in G_i \end{aligned}$$

In practice, we randomly select a data point, compute the gradient and apply the gradient of this data point to the whole cluster. For instance, given two groups (e.g., G_1 and G_2) with the same labels, we need to move at least $min(|G_1|, |G_2|)$ steps to merge two clusters by moving single data points one by one. However, we only need to move once on the group level.

In the same way, we stop moving the data groups when their position are stable:$0.97 < per < 0.99$.

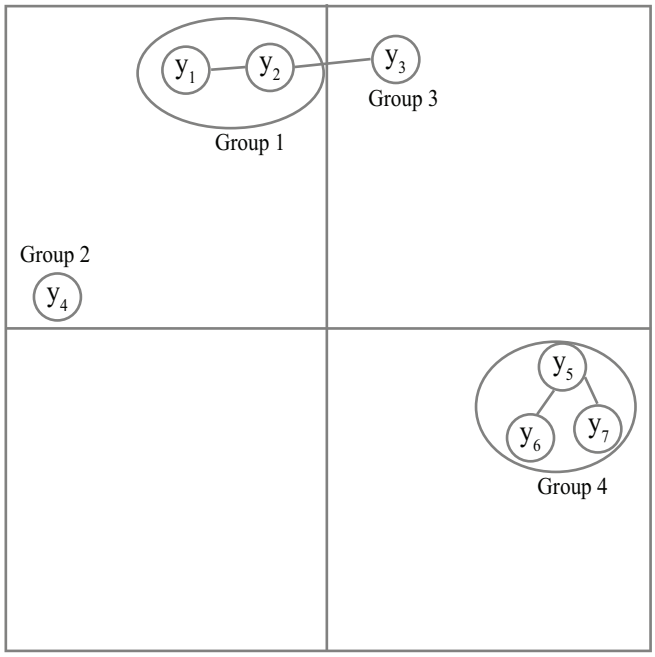


Fig. 2. The data groups are constructed by quadtree and Girvan-Newman algorithm. The data points in each cell have the following characters: local proximity and high dimensional similarity.

4.2.3 Refine layout

There are some data points assigning to a wrong group and move to an inappropriate position. In this stage, we focus on refining the embedding layout by placing these data points to a better position. Therefore, we are expected to move these data points on point-scale. We use original optimization to refine dimensional reduction result.

5 Experiments

In this section, we show the quantitative and qualitative embedding result of our method. We show the result of three experiments. In the first experiment, we study the performance of K-nearest neighbor graph construction and verify the importance of K-nearest neighbor graph accuracy on the quality of embedding. In the second experiment, we compare the efficiency and effectiveness of different visualization algorithms. The third experiment compares the visualization result qualitatively. We conducted all experiments on a single desktop PC with Intel Xeon E5 1660 CPU and 16GB memory.

5.1 Data Sets

We performed experiments on a broad range of data sets and compared our method with state-of-the-art methods.

1) The MNIST[1] data set contains 70,000 grey scale handwriting digital images. Each image contains 28 * 28 = 784 pixels and belongs to one of ten digitals from 0 to 9. Each image is treated as a 784 dimension data point.
2) The fashion-MNIST[2] is a data set consisting of 70,000 grey scale images with labels from 10 fashion product categories. It shares the same image size with MNIST data set. Fashion-MNIST data set is introduced to replace the original MNIST dataset which is too easy for currently machine learning model.
3) The CIFAR-10[3] data set includes 60,000 color images with 32 * 32 resolution. All images are labeled with 10 classes, such airplane, bird, etc. There are 6,000 images in each class.
4) The CIFAR-100 data set is similar with CIFAR-10. There are 100 classes with 600 images each.
5) The street view house numbers (SVHN) data set[4] is a real-world house number image data set obtained from Google Street View images. We use all 630,420 color images in character level format where digits are resized to a resolution of 32 * 32 pixels. The SVHN data set is much harder than MNIST to recognize digits. We employ a pre-trained deep neural network to preprocess the image and extract better data representation for visualization. The neural network architecture contains eight layers: the first seven are convolution layers (with batch normalization, ReLU activations and max-pooling over 3 * 3 regions) and the last one is a softmax layer. This network achieve ?% error on training data, ?% error on test data and ?% error on extra data. We input an 1024 dimensional data into network and the number neurons in each layers is 3 * 32 * 32, 32 * 30 * 30, 32 * 28 * 28, 64 * 12 * 12, 64 * 10 * 10, 128 * 3 * 3, 256 * 1 * 1 and 10. We extract 256-dimensional output of the last convolution layer with as learned representation.
6) The twitter data set[5] contains 1.2M word vectors pre-trained by Glove [21]. Each word is represented by a 200-dimension vector. The distance between two word vectors is an efficient measurement of semantic similarity.

1. http://yann.lecun.com/exdb/mnist/
2. https://github.com/zalandoresearch/fashion-mnist
3. https://www.cs.toronto.edu/ kriz/cifar.html
4. http://ufldl.stanford.edu/housenumbers/
5. https://nlp.stanford.edu/projects/glove/

TABLE 2
Summary of data sets

Data set	Size	Dimension	Class number
MNIST	70,000	784	10
Fashion-MNIST	70,000	784	10
CIFAR-10	60,000	1024	10
CIFAR-100	60,000	1024	100
SVHN	630,420	256	10
Twitter	1,193,514	200	None

5.2 Embedding Result Evaluation

We perform experiments on above data sets and evaluate the performance with the following dimensional reduction algorithms:

- t-SNE [18]: The t-SNE algorithm is the most popular as well as effective method to visualize high-dimension data.
- Barnes-Hut-SNE [28]: The Barnes-Hut-SNE technique accelerate t-SNE from $O(N^2)$ to $O(NlogN)$.
- LargeVis [25]: LargeVis is a large-scale graph visualization method which embedding the kNN graph in high-dimensional space to low-dimensional space.
- Ours: We introduce our method in section ?.

Evaluation We adopt k-NN classifier as the embedding quality metric. We compute the nearest neighborhood classification accuracy based on embedding result. The label of data x_i predicted by k-NN classifier is:

$$\overline{y_i} = arg\max_c \sum_{x_j \in N_k(x_i)} I(y_j = c)$$

Therefore, the accuracy is defined as:

$$A = \frac{1}{N}\sum_{i=1}^{N} I(y_i = \overline{y_i})$$

where x_j is the nearest neighborhood of x_i, y_i and y_j is the label of x_i and x_j. I is identification function. $I(y_i = y_j)$ = 1 when $y_i = y_j$, else $I(y_i = y_j)$ = 0.

5.3 Running Time Comparison

As shown in Table 3, we report the running time of various algorithms on different datesets. We measure the running time of kNN construction (S1), embedding process (S2) as well as total time (Total).

In all cases, our method achieves significant speed up than other methods. For example, on small data set such as MNIST, LargeVis took about 200 seconds while our methods took 35 seconds. For larger-scale data set such as Twitter, our methods is ? times faster than LargeVis and ? times faster than t-SNE. In summary, our methods is more scalable and efficient than previous methods.

5.4 KNN Classifier

We compare the embedding quality of embedding algorithm by using kNN classifier as mentioned in section ?. Figure ? shows the result of classification accuracy by kNN classifier with different k. The KNN classification accuracy result between LargeVis and ours is very similar, indicating that our method achieves a comparable embedding quality. When we construct kNN graph with different k, the classification result in Fig.? shows that kNN graph with small k achieves similar performance too.

5.5 Visualization Comparison

We present the visualization result of LargeVis and our method on MNIST, CIFAR-100 and Twitter data sets in Figure ?. Though we reduce the total number of sampling edges to 1/10, our method achieves comparable result on different data sets.

Figure ? shows the two-dimensional embedding result on MNIST data set. The color of each dot represents the corresponding label. Ten categories of data points are well separated. The 1-NN classifier error of low-dimensional data is 5.5%, which is similar to the error in original space.

Figure ? shows the embedding process of LargeVis and ours. First, similar data points converge in the very early stage in our method. Second, similar small groups are merged into a big group in several steps. However, LargeVis takes about half the time to aggregate similar clusters.

For twitter data set, there is no obvious cluster structure in visualization result (see Figure ?). Some analysis...

6 Conclusion

In this paper, we present an accelerated version of LargeVis by leveraging space partition method. We explore the idea of optimizing based on the local clusters instead of single data points, which significantly speed up LargeVis. We first employ an efficient version of approximate nearest neighbor method to computer the input similarity on high-dimensional space. Then we accelerate the optimization by sharing the gradient between nearby points in low dimensional space. The experiments on a variety of data sets show the efficiency of our method in comparison with LargeVis. Our method is available at https://gitlab.com/Twilightsnow/DimensionReduction.

In the future, we will develop a GPU version of our method to improve the calculation speed. We plan to apply this idea to other embedding methods, such as t-SNE and word2vec. We will focus on accelerating the graph layout using our method.

Appendix A
Proof of the First Zonklar Equation

Appendix one text goes here.

Appendix B

Appendix two text goes here.

Acknowledgments

The authors would like to thank...

This work is supported by ...

TABLE 3
Running Time

	MNIST			F-MNIST			CIFAR-10			CIFAR-100			SVHN			Twitter		
	S1	S2	T	S1	S2	T	S1	S2	T	S1	S2	T	S1	S2	T	S1	S2	
LargeVis	195.271	404.769		140.540	400.085		225.958	374.678		341.133	375.911		2014.039	1993.339				
Ours	47.513	51.22		42.038	51.219		51.04	48.151		74.65	47.444		378.31	406.288				
Speed Up	4.1X	7.9X		3.34X	7.8X		4.4X	7.8X		4.6X	7.9X		5.3X	4.9X				

TABLE 4
My caption

	MNIST				F-MNIST				CIFAR10				CIFAR100				SVHN	
	1	10	30	50	1	10	30	50	1	10	30	50	1	10	30	50	1	10
LargeVis	94.59	96.57			75	80.17			94.2	96.27			53.7	61.3			95.46	97.06
Ours	94.59	96.91	96.86		74.00	80.59	80.89	80.93	94.88	97.03	97.05	97.03	51.50	59.57	60.47	60.48	95.48	97.10

REFERENCES

[1] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural computation*, 15(6):1373–1396, 2003.

[2] HongYun Cai, Vincent W. Zheng, and Kevin Chen-Chuan Chang. A comprehensive survey of graph embedding: Problems, techniques and applications. *CoRR*, abs/1709.07604, 2017.

[3] Trevor F Cox and Michael AA Cox. *Multidimensional scaling*. CRC press, 2000.

[4] Ronald A Fisher. The use of multiple measurements in taxonomic problems. *Annals of human genetics*, 7(2):179–188, 1936.

[5] Cong Fu and Deng Cai. Efanna: An extremely fast approximate nearest neighbor search algorithm based on knn graph. *arXiv preprint arXiv:1609.07228*, 2016.

[6] Michelle Girvan and Mark EJ Newman. Community structure in social and biological networks. *Proceedings of the national academy of sciences*, 99(12):7821–7826, 2002.

[7] Xiaofei He, Deng Cai, Shuicheng Yan, and Hong-Jiang Zhang. Neighborhood preserving embedding. In *Tenth IEEE International Conference on Computer Vision*, volume 2, pages 1208–1213 Vol. 2, Oct 2005.

[8] Xiaofei He and Partha Niyogi. Locality preserving projections. In *Advances in neural information processing systems*, pages 153–160, 2004.

[9] Geoffrey E Hinton and Sam T Roweis. Stochastic neighbor embedding. In *Advances in neural information processing systems*, pages 857–864, 2003.

[10] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

[11] Shihao Ji, Nadathur Satish, Sheng Li, and Pradeep Dubey. Parallelizing word2vec in shared and distributed memory. *CoRR*, abs/1604.04661, 2016.

[12] I. T. Jolliffe. *Principal Component Analysis and Factor Analysis*, pages 115–128. Springer New York, New York, NY, 1986.

[13] Andrej Karpathy, Justin Johnson, and Li Fei-Fei. Visualizing and understanding recurrent networks. *arXiv preprint arXiv:1506.02078*, 2015.

[14] Minjeong Kim, Minsuk Choi, Sunwoong Lee, Jian Tang, Haesun Park, and Jaegul Choo. Pixelsne: Visualizing fast with just enough precision via pixel-aligned stochastic neighbor embedding. *arXiv preprint arXiv:1611.02568*, 2016.

[15] T. Kohonen. The self-organizing map. *Proceedings of the IEEE*, 78(9):1464–1480, Sep 1990.

[16] Joseph B Kruskal. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, 29(1):1–27, 1964.

[17] O. H. Kwon, T. Crnovrsanin, and K. L. Ma. What would a graph look like in this layout? a machine learning approach to large graph visualization. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):478–488, Jan 2018.

[18] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008.

[19] Ahmed Mahfouz, Martijn van de Giessen, Laurens van der Maaten, Sjoerd Huisman, Marcel Reinders, Michael J. Hawrylycz, and Boudewijn P.F. Lelieveldt. Visualizing the spatial gene expression organization in the brain through non-linear similarity embeddings. *Methods*, 73(Supplement C):79 – 89, 2015. Spatial mapping of multi-modal data in neuroscience.

[20] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[21] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[22] N. Pezzotti, B. P. F. Lelieveldt, L. v. d. Maaten, T. Hllt, E. Eisemann, and A. Vilanova. Approximated and user steerable tsne for progressive visual analytics. *IEEE Transactions on Visualization and Computer Graphics*, 23(7):1739–1752, July 2017.

[23] J. W. Sammon. A nonlinear mapping for data structure analysis. *IEEE Transactions on Computers*, C-18(5):401–409, May 1969.

[24] Lawrence K Saul and Sam T Roweis. Think globally, fit locally: unsupervised learning of low dimensional manifolds. *Journal of machine learning research*, 4(Jun):119–155, 2003.

[25] Jian Tang, Jingzhou Liu, Ming Zhang, and Qiaozhu Mei. Visualizing large-scale and high-dimensional data. In *Proceedings of the 25th International Conference on World Wide Web*, pages 287–297. International World Wide Web Conferences Steering Committee, 2016.

[26] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. Line: Large-scale information network embedding. In *Proceedings of the 24th International Conference on World Wide Web*, pages 1067–1077. International World Wide Web Conferences Steering Committee, 2015.

[27] Joshua B Tenenbaum, Vin De Silva, and John C Langford. A global geometric framework for nonlinear dimensionality reduction. *science*, 290(5500):2319–2323, 2000.

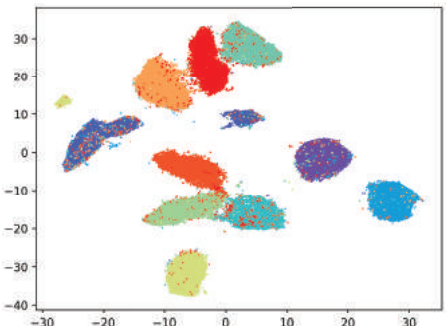
[28] Laurens Van Der Maaten. Accelerating t-sne using tree-based algorithms. *Journal of machine learning research*, 15(1):3221–3245, 2014.

PLACE PHOTO HERE

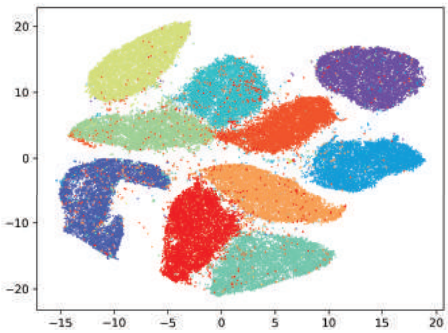
Michael Shell Biography text here.

John Doe Biography text here.

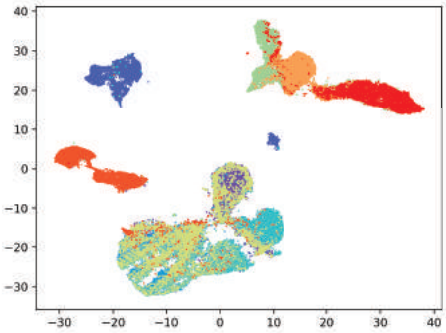
Jane Doe Biography text here.



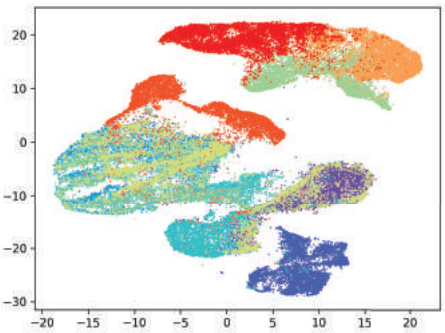
MNIST+LargeVis



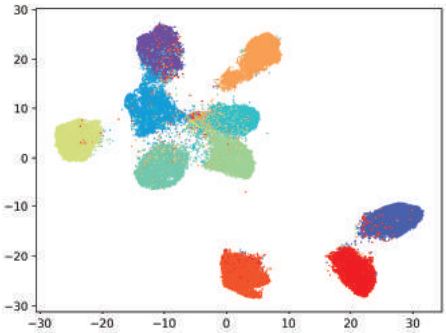
MNIST+Ours



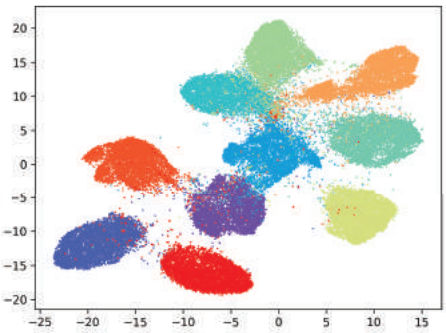
Fashion-MNIST+LargeVis



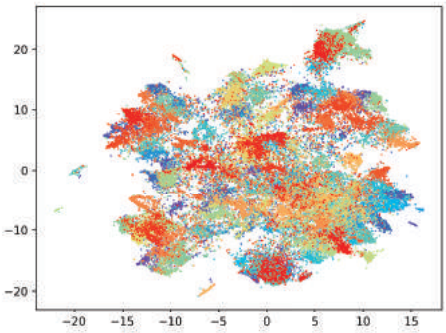
Fashion-MNIST+Ours



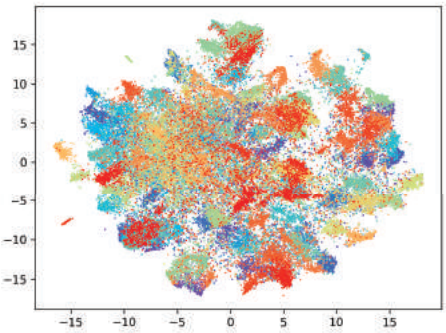
CIFAR 10+LargeVis



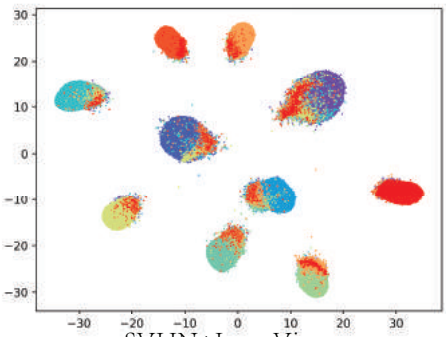
CIFAR 10+Ours



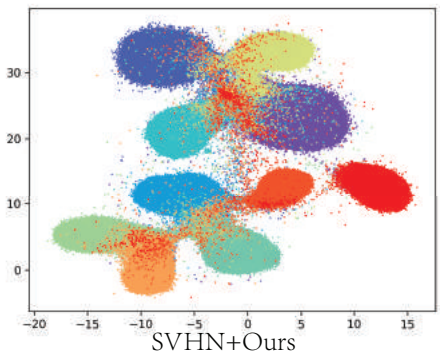
CIFAR 100+LargeVis



CIFAR 100+Ours



SVHN+LargeVis



SVHN+Ours